Laboratorul 10.

Problema 1

Pornind de la codul de mai jos, asigurați faptul că se va apela codul aferent tipului dinamic al parametrului, definind clasele ***Hero***, ***Warrior***, ***Ninja***, ***Rogue*** și ***StormFire***, în mod minimal!

```
public class Binding{
    public static void main(String args[]) {
        Hero h1 = new Warrior(), h2 = new Ninja();
        Hero h3 = new Rogue();
        BadLuck bl = new StormFire();
        bl.execute(h1);
        bl.execute(h2);
        bl.execute(h3);
    }
}

abstract class BadLuck {
    abstract void execute(Hero h);
    abstract void execute(Warrior w);
    abstract void execute(Ninja n);
    abstract void execute(Rogue r);
}
```

Clasele ***Hero*** și ***BadLuck*** sunt clase abstracte!

Problema 2

Să se definească o clasă **generica** ***ArrayMap***, pentru un dicționar realizat din doi vectori (un vector de chei și un vector de valori asociate, obiecte din clasa ***Vector***), care să poată înlocui o clasă ***HashMap*** sau ***TreeMap***. Astfel, această clasă va extinde clasa ***AbstractMap***, suprascriind următoarele metode:

```
public String toString();
public V put(K, V);
public V get(Object );
public Set<K> keySet();
public Collection<V> values();
public Set<Map.Entry<K, V>> entrySet(); // atentie! Se va defini o clasa interna pentru o intrare in dictionar - **//Map.Entry//**
```

Afișați dicționarul folosind System.out.println(dictionar) si apoi folosind un ***Iterator*** pentru a parcurge mulțimea intrarilor generată de metoda ***entrySet***.

Problema 3

Pornind de la clasa abstractă ***AMatrix***, pusă la dispoziție în arhiva laboratorului, implementați clasa ***IntegerMatrix*** care moștenește această clasă abstractă și modelează un tablou bidimensional cu numere întregi. Clasa ***AMatrix*** moștenește clasa ***ArrayList***. Astfel, matricea propriu-zisă este un obiect de tip ***ArrayList*** care conține elemente de tip ***ArrayList***. Clasa va conține metode pentru următoarele operații: afișarea matricei, adunarea a două matrice, și metoda sum pentru a aduna două elemente!.

```
//afisare
public String toString();
//sum
public Integer sum(Integer obj1, Integer obj2)
//adunare
public AMatrix addition(AMatrix m);
```

Folosiți iteratori pentru parcurgerea colecțiilor sau bucle for each!

Problema 4

Definiți clasa ***GenericListMethods*** care să implementeze interfața, pusă la dispoziție în arhiva laboratorului, ***GenericInterface***. Această interfață conține operații care prelucrează o listă, cu elemente de tip ***Comparable***.

Metoda ***removeDuplicates*** primește ca parametru un obiect de tip ***ArrayList*** și transformă lista într-on mulțime, eliminând duplicatele. Metoda ***max*** are ca parametru tot un ***ArrayList*** și returnează elementul maximal din listă. Ultima metodă conținută de interfață, ***binarySearch***, este folosită pentru a determina poziția unei valori într-o listă ordonata, uzitând pentru aceasta algoritmul de căutare binară, detaliat în blocul de cod de mai jos.

```
int BinarySearch(v, start, end, x) {
    //conditia de oprire (x nu se afla in v)
    if (start > end)
        return -1;
    //etapa divide
    int mid = (start + end) / 2;
    //etapa stapaneste
    if (v[mid] == x)
        return mid;
    if (v[mid] > x)
        return BinarySearch(v, start, mid-1, x);
    if (v[mid] < x)
        return BinarySearch(v, mid+1, end, x);
}
```

Arhivă laborator

poo/laboratoare/09.txt · Last modified: 2022/11/30 11:35 by carmen.odubasteanu